Short Paper: Mechanized Proofs of Masking Security

Roberto Blanco, Christian Doczkal, Jakob Feldtkeller, Tim Güneysu, Cătălin Hriţcu

Short Paper: Mechanized Proofs of Masking Security

Roberto Blanco[1] Christian Doczkal[1] Jakob Feldtkeller[2] Tim Güneysu[2] Cătălin Hriţcu[1]

[1] MPI-SP, Bochum, Germany [2] Ruhr University Bochum, Germany

Abstract

Among the many threats that side channels pose to the security of computer systems, those that exploit physical access are among the most insidious. A general countermeasure against these attacks is masking, a form of secret sharing applied to sensitive data. While these techniques are effective and widespread, implementing them correctly typically involves complex informal reasoning, where even subtle mistakes can invalidate the security guarantees they potentially offer. We present our vision and initial work to provide rigorous formal assurances and sound reasoning principles for circuit designers by developing a mechanization framework for masked circuits that can be used to reason about the security of gadgets and their composition. Our work is done on top of the EasyCrypt proof assistant.

1 Masking Security

Side-channel attacks are significant and pervasive threats to the security of computer systems. Among these, physical attacks take advantage of access to the hardware to exfiltrate secrets. Implementations of crypto primitives as hardware circuits and the secret cryptographic material manipulated by these are notable targets for these kinds of attacks. To give any formal account of security against physical side channels, a threat model is necessary. The foundational attacker in this domain is given by the probing model of Ishai et al. [12], where the attacker is able to select up to a given number t of wires in the circuit and observe their values. A circuit is t-probing secure if no combination of up to t probes reveals anything about any sensitive value. The parameter t represents the *security order* of the property. More sophisticated attackers are also defined in the literature [6, 9, 10].

Among the countermeasures devised to hinder the exploitation of physical side channels on hardware circuits, masking stands as one of the most widely used. A masked circuit is one where each piece of secret data x has been decomposed into a number d of randomized, uniformly distributed shares, $x_1, \ldots, x_d$ such that the original value can be reconstructed iff all of the shares are at hand: $x = x_1 \oplus \ldots \oplus x_d$. Wires in masked circuits carry shares of secret inputs, outputs and intermediate computations, as well as random bits used to conceal the latter. The number of shares, given by the parameter d, is related to the security guarantees offered by a masked circuit, although establishing a connection between the two is complex and error-prone.

Given a circuit, it is possible in general to obtain an equivalent t-probing secure version by applying a masking transformation where the number of shares satisfies $d > t$, and reproducing the original computation on those masked shares. In practice, a complex circuit will be divided into smaller "gadgets," representing, e.g., logic or arithmetic operations, protected individually by masking and then assembled together. However, the combination of probing secure gadgets into larger circuits is not necessarily secure. This has led to the definition of stronger compositional security criteria, like Strong Non-Interference (SNI) [3] and Probe-Isolating Non-Interference (PINI) [8].

Given the difficulty of manually verifying the security of even small gadgets at moderate orders, a more rigorous treatment of the verification problem becomes essential. Most existing verification tools apply automated reasoning techniques to a selection of probing models and security properties [1, 7, 13]. These tools are not easily extensible, and do not scale beyond the verification of small individual masked gadgets at given, fixed orders. Despite the recent pioneering work of Barthe et al., [2], the potential of semi-interactive verification tools has remained largely unexplored.

We envision that by developing a formal framework for mechanized proofs of masking security in a semi-interactive proof assistant, we can faithfully reproduce and facilitate the kinds of security reasoning carried out by expert circuit designers, increasing confidence and avoiding the kinds of vulnerabilities that have marred masked designs in the past (e.g., [11]. Our vision is one where mechanized proofs and informal reasoning and design patterns are in close correspondence, and moreover specifications and proofs are connected to secure implementations "all the way down."

2 Mechanized Proofs

We now outline our proposed approach to formalize masked circuits, attacker models, and security definitions in two layers: a first layer to reason about the security of individual gadgets (§2.1), and a second layer to reason about the compositional security of circuits formed of smaller gadgets (§2.2). We have started developing the basic building blocks of the framework (definitions on both layers) and also experimenting with this by developing as a case study for the first layer the core security proofs for a gadget exemplar (ISW multiplication [12]). For this we use EasyCrypt [4], a semi-interactive proof assistant for the specification and development of cryptographic proofs using probabilistic relational Hoare logic (pRHL) [5], whose judgments we use to express the security properties of interest clearly and naturally.

2.1 Circuit Security

For the first layer, a natural choice, which we use in our experiments, is to encode masked circuits as procedures in EasyCrypt's probabilistic While language [5]. Masked inputs and outputs are represented as vectors of randomized shares, intermediate computations are stored in vectors and matrices, and random bits are obtained using the language's built-in generation primitives. Consider ISW multiplication [12], our initial case study. A slightly simplified model of this gadget can be given as an EasyCrypt procedure as follows:

```
proc mult(a b : M.vector) : M.vector = {
  var w, u, r : matrix;
  var c : M.vector;
  r <$ dcond (dmatrix dbool)
        (fun m => msym m /\ diagf m);
  w <- offunm (fun i j => a.[i] /\ b.[j]);
  u <- offunm (fun i j => w.[i, j] ^ r.[i, j]);
  c <- offunv (fun i => BBA.bigi predT
                (fun j => u.[i, j]) 0 (d + 1));
  return c; }
```

In this programmatic representation, we can approximate probe locations as indexes in each vector and matrix. To study the security of these procedural circuits, we can wrap their code in a leakage procedure that performs the computation of the circuit and extracts the values associated with its probed variables. This is specific to each leakage model.

```
proc probe(a b : M.vector,
           obs : ProbeSet) : LeakageSet = {
  ... (* compute gadget *)
  return (leakage obs a b r w u c); }
```

For example, if the probe set contains locations $\ldots, w_{1,2}, \ldots$, and the gadget runs with $a_1 = 0$, $b_2 = 1$, the corresponding output will leak $\ldots, w_{1,2} = (a_1 \wedge b_2) = (0 \wedge 1) = 0, \ldots$.

A proof of probing security boils down to a simulation between the probes of two circuits. Technically, given a gadget (the first circuit) and a valid set of probes, we can construct a simulator (the second circuit) that isolates those shares of the inputs that influence the probed values. The second circuit uses this subset of shares to effectively simulate the probes in the original gadget, that is, the probability distributions of the values read by each probe are identical in the original gadget and in the simulator. The corresponding EasyCrypt lemma is a pRHL equivalence on the outputs of the leakage procedures for the gadget and its simulator (`={res}`), starting from identical inputs `a`, `b` and well-formed probes `obs` (i.e., arbitrary-order while satisfying $t < d$):

```
lemma ISWsec (probes : ProbeSet) :
  equiv[Gadget.probe ~ Simulator.probe :
    wf_ProbeSet probes /\
    ={a, b, obs} /\ obs{1} = probes ==> ={res}].
```

The goal of the proof is making explicit the probe simulation argument for an arbitrary probe set, and the key step involves establishing an adequate coupling between the randomly sampled values used to conceal the sensitive data flowing through the gadgets. Although formal proofs are much more detailed than paper proofs, most of this process is a prime target for automation. However, a technically interesting point is the disconnect between the sources of randomness in a gadget and its simulator. Since we are only interested in the values leaked by a fixed (but arbitrary) set of probes, a procedure in standard style will oversample and rearrange randomness in complex patterns, obscuring the coupling that constitutes the core of the proof. To expose this core, we can perform a series of game hops to obtain equivalent circuits phrased in terms of lazy oracles, which only sample those random bits that become part of the probed values. Precise couplings are developed at this level.

We note that these proof principles extend naturally to more powerful attackers than those of the standard probing model, easily capturing other hardware issues such as glitches [10]. Different security properties will result in slightly different proofs, but we anticipate that the same proof principles should apply to a variety of compositional criteria built on top of the different threat models.

2.2 Gadget Composition

Once we have a library of masked gadgets, whose security is proved once and for all, we will want to use them to build larger circuits, and to combine their security proofs into security proofs for the resulting circuits. Because t-probing security is not compositional, in practice we wish to prove stronger properties, like SNI or PINI [3, 8], that add support for secure composability of gadgets under easily checkable and automatable conditions. We expect those proofs to largely follow the structure of existing proofs. Mechanizing the second layer will involve formalizing the composition theorems of these properties and some proof automation.

The encoding of gadgets-as-procedures is both intuitive and close to EasyCrypt reasoning principles; a drawback is that the desired composition theorems need to be stated at the meta level. At the object level, a general theorem of (say) SNI would need to universally quantify over families of procedures (and the modules that contain them), but this is not possible because procedures are not first-class values in EasyCrypt. What we will be able to do at the object level is to separately model gadgets and circuits as data structures (such as standard netlists), and security properties as logic predicates on those structures. Using this representation, we can define and formalize theories of secure composability that quantify over gadget structures. While mechanically establishing a connection between gadgets-as-procedures (where security proofs of individual gadgets are carried out) and gadgets-as-data-structures (where those proofs are composed) is not immediately clear, we can envision a compiler translating between EasyCrypt data gadgets and procedures, as well as a proof relating the two representations.

Acknowledgments. We thank the anonymous reviewers at PLAS'23 for their thoughtful comments. This work was supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Excellence Strategy of the German Federal and State Governments – EXC 2092 CASA - 390781972.

References

[1] G. Barthe, S. Belaïd, G. Cassiers, P. Fouque, B. Grégoire, and F. Standaert. maskverif: Automated verification of higher-order masking in presence of physical defaults. In K. Sako, S. A. Schneider, and P. Y. A. Ryan, editors, *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Proceedings, Part I*. 2019.

[2] G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, and P. Strub. Verified proofs of higher-order masking. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part I*. 2015.

[3] G. Barthe, S. Belaïd, F. Dupressoir, P. Fouque, B. Grégoire, P. Strub, and R. Zucchini. Strong non-interference and type-directed higher-order masking. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016.

[4] G. Barthe, F. Dupressoir, B. Grégoire, C. Kunz, B. Schmidt, and P. Strub. Easycrypt: A tutorial. In A. Aldini, J. López, and F. Martinelli, editors, *Foundations of Security Analysis and Design VII - FOSAD 2012/2013 Tutorial Lectures*. 2013.

[5] G. Barthe, B. Grégoire, and S. Zanella-Béguelin. Formal certification of code-based cryptographic proofs. In *Proceedings of the 36th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2009, Savannah, GA, USA, January 21-23, 2009*, 2009.

[6] A. Battistello, J. Coron, E. Prouff, and R. Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In B. Gierlichs and A. Y. Poschmann, editors, *Cryptographic Hardware and Embedded Systems - CHES 2016 - 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings*. 2016.

[7] S. Belaïd, D. Mercadier, M. Rivain, and A. R. Taleb. Ironmask: Versatile verification of masking security. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*. 2022.

[8] G. Cassiers and F. Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[9] A. Duc, S. Dziembowski, and S. Faust. Unifying leakage models: From probing attacks to noisy leakage. In P. Q. Nguyen and E. Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*. 2014.

[10] S. Faust, V. Grosso, S. M. D. Pozo, C. Paglialonga, and F. Standaert. Composable masking schemes in the presence of physical defaults & the robust probing model. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):89–120, 2018.

[11] J. D. Golic and C. Tymen. Multiplicative masking and power analysis of AES. In B. S. K. Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*. 2002.

[12] Y. Ishai, A. Sahai, and D. A. Wagner. Private circuits: Securing hardware against probing attacks. In D. Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Proceedings*. 2003.

[13] D. Knichel, P. Sasdrich, and A. Moradi. SILVER - statistical independence and leakage verification. In S. Moriai and H. Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Proceedings, Part I*. 2020.